RESEARCH ARTICLE

Deep learning based searching approach for RDF graphs

Hatem Soliman *

College of Computer Science & Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

* hatem@nuaa.edu.cn

Abstract

The Internet is a remarkably complex technical system. Its rapid growth has also brought technical issues such as problems to information retrieval. Search engines retrieve requested information based on the provided keywords. Consequently, it is difficult to accurately find the required information without understanding the syntax and semantics of the content. Multiple approaches are proposed to resolve this problem by employing the semantic web and linked data techniques. Such approaches serialize the content using the Resource Description Framework (RDF) and execute the queries using SPARQL to resolve the problem. However, an exact match between RDF content and query structure is required. Although, it improves the keyword-based search; however, it does not provide probabilistic reasoning to find the semantic relationship between the queries and their results. From this perspective, in this paper, we propose a deep learning-based approach for searching RDF graphs. The proposed approach treats document requests as a classification problem. First, we preprocess the RDF graphs to convert them into N-Triples format. Second, *bag-of-words* (BOW) and *word2vec* feature modeling techniques are combined for a novel deep representation of RDF graphs. The attention mechanism enables the proposed approach to understand the semantic between RDF graphs. Third, we train a convolutional neural network for the accurate retrieval of RDF graphs using the deep representation. We employ 10-fold cross-validation to evaluate the proposed approach. The results show that the proposed approach is accurate and surpasses the state-of-the-art. The average *accuracy*, *precision*, *recall*, and *f-measure* are up to *97.12%*, *98.17%*, *95.56%*, and *96.85%*, respectively.

OPEN ACCESS

Citation: Soliman H (2020) Deep learning based searching approach for RDF graphs. PLoS ONE 15 (3): e0230500. https://doi.org/10.1371/journal.pone.0230500

Editor: Tao Song, Polytechnical Universidad de Madrid, SPAIN

Received: October 17, 2019

Accepted: March 3, 2020

Published: March 23, 2020

Data Availability Statement: The data underlying this study is available from the the DBpedia dataset (https://wiki.dbpedia.org/data-set-30). DBpedia 2016-10 release contains 13 billion 195 pieces of information out of which 1.7 billion were extracted from the English 196 edition of Wikipedia. We use only 1.7 billion RDF triples (English version) to 197 evaluate the proposed approach; however, we ignore all syntactically invalid triples.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

1 Introduction

The digital age arrives with a set of challenges for the Web due to the abundance of information. In today's modern society, people capture, upload, store, and digitalize almost every activity of daily life routine over the Web. Nowadays, communication devices have the capacity to connect to the internet independently and contain sensors that are spreading useful information without users' intervention. Consequently, the data is increasing daily and resulting in *information overload*. Searching such data had driven to the development of the linked data and Semantic Web. It considers the *machine processable* metadata [1] that enhances information flow and can relate data from distributed data sources to make data meaningful. This mash-up of data introduced the phrase *Web 3.0*. Building links between distributed data sources is essential to Web 3.0 which is achieved using RDF. RDF resources consist of RDF triples where each triple contains a subject *s* that has property *p* with value *o* [2, 3]. Consequently, a RDF triple may be viewed as a representation of an atomic *fact* or a *claim* [4].

The analogous datasets may be defined as *Linked Data* [5] that can be summarized as being "*simply about using the Web to create typed links between data from different sources*". Linked data combine entities from different sources/locations to crawl them as a data-space due to its connected links [5, 6]. This idea motivates this study to access the required information from distributed sources and build links that help in searching. RDF triples allow entities to be queried and linked together. The existing studies use RDF and SPARQL to serialize the content and to execute the queries for searching, respectively.

RDFs are massive in size and crucial; therefore it is not easy to extract information for an ordinary user. Although, linked data and SPARQL provides a significant improvement in search methods. However, the complexity criteria (similar triples by RSFS and OWL rules) and the usability criteria (the human effort) are required to read and learn RDF data [7]. For example, SPARQL queries require structure accuracy to extract RDF elements. Such queries do not allow the statistic analysis to evaluate the query against the RDF content; e.g., features of a basket may not be enough as input to identify the *online shopping basket*. Many approaches have been proposed for achieving this kind of searching from RDF data using linked data and SPARQL [8–19]. Notably, such approaches respond to queries with an exact match rather than estimating the similarity within the RDF content that leads to the original motivation for the work in this paper. On the other hand, Hadi *et al.* [20] exploited a machine learning approach to search RDF graphs. Although their approach works on statistical estimation, it does not consider semantic relationships while searching RDF graphs and requires significant improvement.

To this end, a deep learning-based searching approach is proposed for RDF graphs. In this regard, we first reuse the history-data of *DBpedia* documents. Second, we preprocess the extracted RDF graphs using the W3C validation service. Third, we concatenate *BOW* and *word2vec* feature modeling techniques for attention-based recurrent neural network novel deep representation of RDF graphs. The attention mechanism enables the proposed approach to understand the semantic between RDF graphs. Fourth, we train a convolutional neural network for the accurate retrieval of RDF graphs using the deep representation. Finally, a convolutional neural network is trained to predict the retrieval of RDF graphs. And the proposed approach is evaluated using a 10-fold cross-validation technique on the given dataset. The evaluation results show the accuracy of the proposed approach. The average *accuracy*, *precision*, *recall*, and *f-measure* are up to *97.12%*, *98.17%*, *95.56%*, and *96.85%*, respectively.

The main contributions of this study are as follows:

- An approach based on a convolutional neural network is proposed for searching RDF graphs. To the best of our knowledge, we are the first to exploit a deep learning algorithm in retrieval prediction of RDF graphs.
- Evaluation results of the proposed approach on the given dataset show that the proposed convolutional neural network-based approach is accurate and surpasses the state-of-the-art.

The rest of the paper is organized as follows: Section II presents the proposed approach. The evaluation process and results of the proposed approach are described in Section III. Section IV explains the threats. Section V and Section VI present the related work and conclusion, respectively.

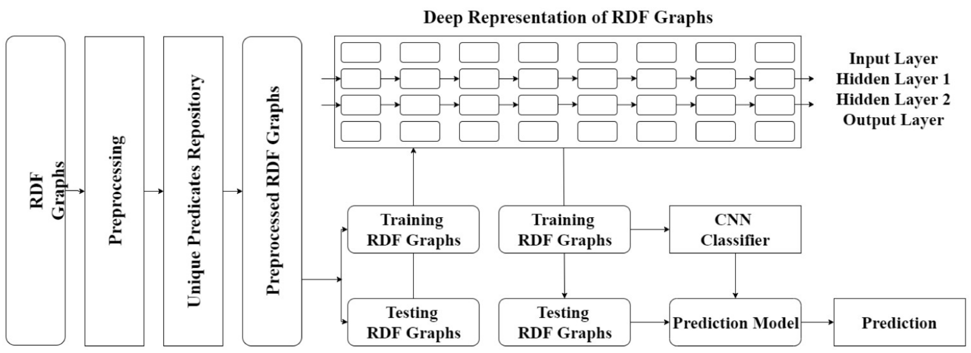


Fig 1. Overview of the proposed approach.

https://doi.org/10.1371/journal.pone.0230500.g001

2 Approach

2.1 Overview

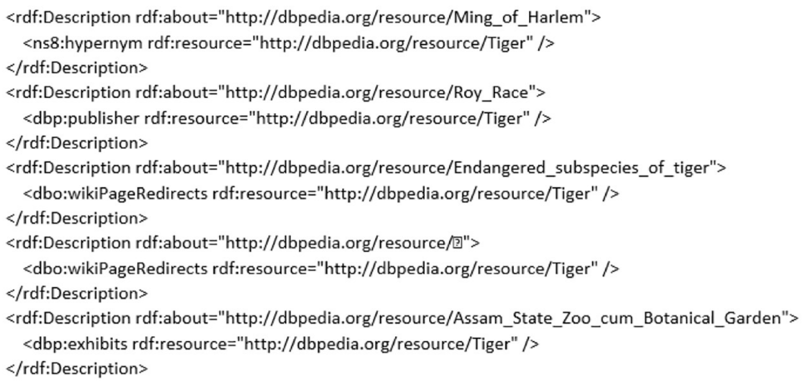
Fig 1 illustrates an overview of the convolutional neural network based searching for RDF graphs. The proposed approach performs RDF graph retrieval prediction as follows:

1. We reuse the history-data of RDF graphs as training data.
2. We apply the W3C validation service to RDF graphs for preprocessing.
3. We concatenate *BOW* and *word2vec* feature modeling techniques for a novel deep representation of RDF graphs.
4. A convolutional neural network is trained to anticipate the retrieval of RDF graphs. We pass the deep representation to the classifier as input that predicts the retrieval of RDF graphs.

The following sections introduce the key steps of the proposed approach.

2.2 Illustrating example

We consider an example to demonstrate how the proposed approach anticipates the retrieval of RDF graphs. An excerpt of RDF graph taken from *DBpedia* is presented in Fig 2. The details



```
<rdf:Description rdf:about="http://dbpedia.org/resource/Ming_of_Harlem">
  <ns8:hypernym rdf:resource="http://dbpedia.org/resource/Tiger" />
</rdf:Description>
<rdf:Description rdf:about="http://dbpedia.org/resource/Roy_Race">
  <dbp:publisher rdf:resource="http://dbpedia.org/resource/Tiger" />
</rdf:Description>
<rdf:Description rdf:about="http://dbpedia.org/resource/Endangered_subspecies_of_tiger">
  <dbo:wikiPageRedirects rdf:resource="http://dbpedia.org/resource/Tiger" />
</rdf:Description>
<rdf:Description rdf:about="http://dbpedia.org/resource/⍰">
  <dbo:wikiPageRedirects rdf:resource="http://dbpedia.org/resource/Tiger" />
</rdf:Description>
<rdf:Description rdf:about="http://dbpedia.org/resource/Assam_State_Zoo_cum_Botanical_Garden">
  <dbp:exhibits rdf:resource="http://dbpedia.org/resource/Tiger" />
</rdf:Description>
```

Fig 2. An example of RDF graph.

https://doi.org/10.1371/journal.pone.0230500.g002

on how the proposed approach deals the illustrating example are given in the following sections.

2.3 Problem definition

A RDF d is a graph from a set of RDF graphs (D) which can be formalized as,

$$d = \ < t_1, t_2, \ldots, t_n > \tag{1}$$

where, t_1, t_2, . . ., t_n represents the n number of triples in a RDF graph, and each triple consists of *subject*(s), *predicate*(p), and *object*(o).

$$d_e = \ < t_{e_1}, t_{e_2}, \ldots, t_{e_n} > \tag{2}$$

where,

d_e = a complete example of a RDF graph represented in Fig 2,

t_{e_1} = first triple of the example,

t_{e_2} = second triple of the example,

.

.

t_{e_n} = last triple of the example.

The proposed approach takes the problem of searching of RDF graphs as classification problem and predicts whether a RDF graph will be retrieved or not. The retrieval anticipation of a new RDF graph d can be defined as a function f.

$$c = f(d) \quad c \in \{hit \ or \ miss\}, \quad d \in D \tag{3}$$

where, c, f, d, and D are predefined classification (*hit* or *miss*), classification function of retrieval anticipation, a RDF graph, and a set of RDF graphs, respectively.

2.4 Preprocessing

We preprocess each of the RDF graph using the *W3C Validation Service*. We load each collected RDF graph using *Apache Jena API* (http://jena.apache.org/) to validate its syntax. Then, the validated RDF graph is loaded into the model that transforms the RDFs from *serialization* format to *N-Triples* format. The preprocessing of a RDF graph can be formalized as,

$$d = \ < t_1', t_2', \ldots, t_n' > \tag{4}$$

where, $t_1', t_2', \ldots, t_n'$ are preprocessed n triples of the RDF graph d.

For the motivating example presented in Section 2.2, a RDF graph d_e after preprocessing can be formalized as,

$$d_e = \ < t_{e_1}', t_{e_2}', \ldots, t_{e_n}' > \tag{5}$$

where, from the excerpt of preprocessed RDF graph is presented in Fig 3, $t_{e_1}', t_{e_2}', \ldots, t_{e_n}'$ represent the n triples (separated with a dot (.)) of the preprocessed RDF graph, respectively.

2.5 Deep feature representation

A *BOW* representation of each RDF graph provides a boolean (0 or 1) array or a term frequency array using all repository terms [21] (shown in Fig 4) and does not incorporate the semantic similarity among terms. Moreover, problems like high dimensionality and sparse data are observed in the bag-of-n-words feature representation [22]. To this end, a neural

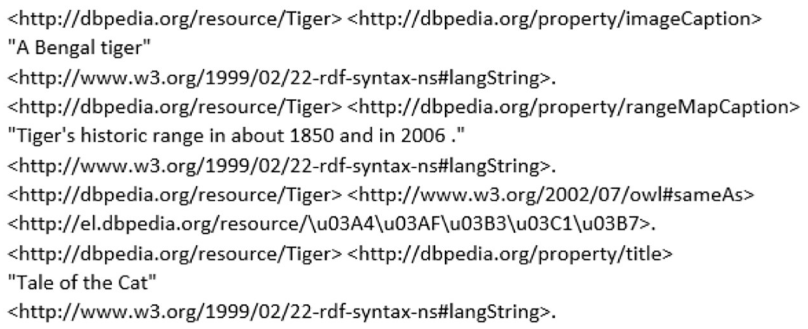

<http://dbpedia.org/resource/Tiger> <http://dbpedia.org/property/imageCaption>
"A Bengal tiger"
<http://www.w3.org/1999/02/22-rdf-syntax-ns#langString>.
<http://dbpedia.org/resource/Tiger> <http://dbpedia.org/property/rangeMapCaption>
"Tiger's historic range in about 1850 and in 2006 ."
<http://www.w3.org/1999/02/22-rdf-syntax-ns#langString>.
<http://dbpedia.org/resource/Tiger> <http://www.w3.org/2002/07/owl#sameAs>
<http://el.dbpedia.org/resource/\u03A4\u03AF\u03B3\u03C1\u03B7>.
<http://dbpedia.org/resource/Tiger> <http://dbpedia.org/property/title>
"Tale of the Cat"
<http://www.w3.org/1999/02/22-rdf-syntax-ns#langString>.

Fig 3. Preprocessed example of RDF graph.

https://doi.org/10.1371/journal.pone.0230500.g003

network-based features representation model (*word2vec*) is proposed to learn and understand the semantic relationship between terms (predicates in our case) [23]. However, *word2vec* only considers semantics of individual terms rather than a sequence of words. Notably, a significant improvement is required to combine the sequence of terms, the syntax of terms, the semantic relationship among terms. In this perspective, a deep representation of RDF graphs is proposed. Fig 5 illustrates an overview of the deep representation of RDF graphs. The long short term memory (LSTM) cells are exploited [24] as a memory unit in the hidden layer that resolve the vanishing gradient problem [25]. LSTM cells can memorize the sequence of terms in both forward direction and backward direction.

The construction of deep representation involves the extraction of $|U|$- dimensional representation (*BOW*) using predicate repository, the learning of $|V|$-dimensional representation *word2vec* using $|U|$-dimensional representation, and the learning of LSTM cells (deep representation) using $|S|$-dimensional representation. This process returns the $|D|$-dimensional representation of the given RDF graph. The $|D|$-dimensional representation has a sequence network (recurrent neural network) that contains a hidden layer with n hidden units ($h = h_1, h_2, \ldots\ldots, h_n$). The recurrent neural network takes $|V|$-dimensional representation ($y = y_1, y_2, \ldots\ldots, y_n$) as an input and returns a $|D|$-dimensional representation ($z = z_1, z_2, \ldots\ldots, z_n$). Every h transforms the previous state s_{i-1} and a term y_i into the next state s_i and output word z_i. Every hidden unit repeatedly performs the function f in recurrent neural network:

$$f : \{s_{i-1}, y_i\} \rightarrow \{s_i, z_i\} \tag{6}$$

where each s_i has the information of i^{th} term in h, whereas the output z_n of h_n represents the complete RDF graph.

Additionally, an attention mechanism is employed to learn from the predicates of the RDF graph. An attention vector with the weighted summation of all outputs z_i can be formalized as,

$$a_n = \sum_{i=1}^{n} \alpha_i z_i \tag{7}$$

where α_i is the weight of each word y_i that defines the importance of y_i for classification. A

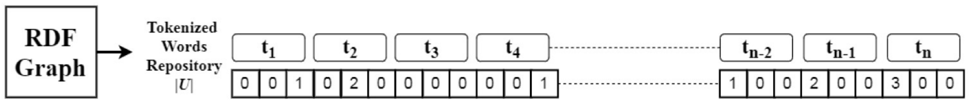


Fig 4. Traditional feature representation model.

https://doi.org/10.1371/journal.pone.0230500.g004

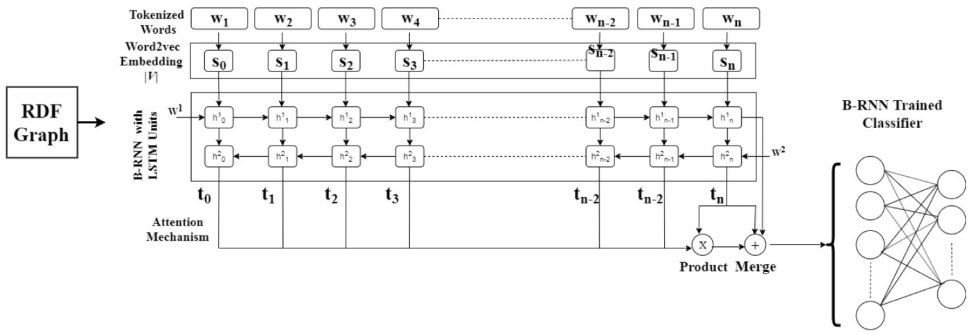


Fig 5. Deep representation model.

https://doi.org/10.1371/journal.pone.0230500.g005

bidirectional recurrent neural network learns representation with input word sequence (forward and backward). A complete deep representation of d can be formalized as,

$$d = z_n + a_n + a_n + z_n \tag{8}$$

where, + represents the concatenation of vectors.

The hyper-settings of the proposed deep representation model as follows: 300 LSTM units, 0.2 dropout probability, 0.001 learning rate, and *binary cross-entropy* based loss function with *Adam optimizer*. We set 100 epochs for the training. Notably, the proposed model is implemented in *Python Keras Library* [26]. To the best of our knowledge, we are the first to apply deep representation to learn the RDF graph representation. We use deep representation to train a convolutional neural network for the retrieval anticipation of RDF graphs.

2.6 Deep learning classifier

Fig 6 illustrates an overview of deep learning classifier. We leverage the convolutional neural network for retrieval prediction of RDF graphs. We select the convolutional neural network because of the following reasons: 1) its deep semantic relationship learning capabilities among words [27]; 2) it avoids the gradient problem of recurrent neural network [28] by applying different filter sizes.

To train the convolutional neural network, the deep representation is forwarded to convolutional neural network that contains 3 layers of CNN, filter 128, kernel size 1, loss function *binary-crossentropy*, and activation *tanh*. Then, the output of the convolutional neural network is passed to a flatten layer [29] that returns a 1-dimensional vector. Finally, the dense layer connects the neurons between layers and the output layer returns the retrieval prediction of RDF graphs.

3 Evaluation

This section defines the research questions to evaluate the proposed approach, explains how RDF graphs are collected, presents the metrics and evaluation process of the proposed approach, and discusses the results while answering the research questions.

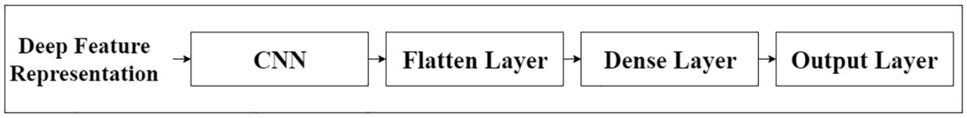


Fig 6. Overview of the deep learning-based classifier.

https://doi.org/10.1371/journal.pone.0230500.g006

3.1 Research questions

The proposed approach is evaluated by investigating the following research questions:

- RQ1: How accurate the proposed approach in retrieval prediction of RDF graphs?
- RQ2: Does the proposed classifier outperform other machine/deep learning classifiers in retrieval prediction of RDF graphs?
- RQ3: Does features' preprocessing influence in predicting the retrieval of RDF graphs?

The *RQ1* examines the accuracy of the proposed approach. In this perspective, the proposed approach is compared with the state-of-the-art approaches: a graph-based retrieval of RDFs (GRSearch) [30] and machine learning-based retrieval of RDFs (MLSearch) [20]. We also compare the proposed approach with the two baseline algorithms: *random prediction algorithm* and *zero-rule algorithm* to double-check the performance of the proposed approach.

The *RQ2* compares the performances of different machine/deep learning classifiers to reveal whether the proposed approach outperforms other machine/deep learning classifiers in retrieval prediction of RDF graphs.

The *RQ3* examines the influence of the features' preprocessing. In this perspective, we compute and compare the performance of the proposed approach with and without preprocessing.

3.2 Dataset

We collect the *DBpedia* dataset (https://wiki.dbpedia.org/data-set-30). DBpedia 2016-10 release contains 13 billion pieces of information out of which 1.7 billion were extracted from the English edition of Wikipedia. We use only 1.7 billion RDF triples (English version) to evaluate the proposed approach; however, we ignore all syntactically invalid triples, as mentioned in Section 2.4. Note that we divide the data into four different search categories: *Triple-pattern requests with multiple responses; e.g., British actors and their birth regions*, *Extended triple-pattern requests with multiple responses; e.g., Movies having award-winning feminist actors*, *Triple-pattern requests with zero responses; e.g., MIT graduates born in Steve Jobs's death place*, and *Extended triple pattern requests with zero responses; e.g., People who influenced by Egyptian writers* to evaluate the proposed approach.

3.3 Process and metrics

Algorithm 1 shows the process to compute the best classifier (*CNN*) as mentioned in Section 2. Algorithm 1 consists of three parts. In the first part (Line 1), we set cross-validation (sometimes called rotation estimation) [31] *M* on *D*. We divide *D* into ten segments notated as $M_i (i = 1, 2, \ldots, 10)$. We subtract the RDF graphs that belong to M_i and mark them as testing RDF graphs *Test*, and the remaining RDF graphs are marked as training RDF graph *Train*. In the second phase (Lines 2-11), we apply the *M*-fold cross-validation and train/test the classifiers (MNB, LR, RF, SVM, LSTM, and CNN). For each iteration of cross-validation, we first separate the training dataset *Train* and testing dataset *Test* (Line 3). Then, we train the classifiers with *Train* and evaluate each classifier with *Test* (Lines 4-10). In the last phase (Lines 12-13), we compute and compare the metrics (*accuracy*, *precision*, *recall*, and *F1*) of each classifier, and return the best classifier.

Algorithm 1 Identification of Best Machine/Deep Learning Algorithm for the Proposed Approach

```
Input: DBpedia RDF graphs $D$
Output: Best machine learning classifier for Searching $\subseteq D$ from $D$
1: $M_i (i = 1, 2, 3, \ldots\ldots, 10)$; $Train = M_j$; $Test = M_i$; $j \neq i$
```

```
2: for each cross-validation m from M do
3:   $Train_i = \bigcup_{i\in[1,10]\wedge j\neq i} M_j$
4:   Train a Multinomial Naive Bayes (MNB) classifier with training data Train.
5:   Train a Logistic Regression (LR) classifier with training data Train.
6:   Train a Random Forest (RF) classifier with training data Train.
7:   Train a Support Vector Machine (SVM) classifier with training data Train.
8:   Train a Long Short Term Memory (LSTM) classifier with training data Train.
9:   Train a Convolutional Neural Network (CNN) classifier with training data Train.
10:   Take the trained MNB, LR, RF, SVM, LSTM, and CNN for retrieval prediction of each RDF graph with testing data Test.
11: end for
12: Calculate the accuracy, precision, recall, and F1 of MNB, LR, RF, SVM, LSTM, and CNN.
13: Compare their predicted value with actual value.
14: return Best Classifier.
```

The selected metrics are commonly adopted metrics for the performance evaluation of classification algorithms [32–37]. Therefore, we calculate the retrieval related *accuracy*, *precision*, *recall*, and *f-measure* for the performance evaluation of the proposed approach on the given RDF graphs that can be defined as,

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$Pre = \frac{TP}{TP + FP} \tag{10}$$

$$Rec = \frac{TP}{TP + FN} \tag{11}$$

$$F1 = \frac{2 * Pre * Rec}{Pre + Rec} \tag{12}$$

where, *Acc*, *Pre*, *Rec*, and *F1* represent the *accuracy*, *precision*, *recall*, and *f-measure* of the proposed approach in retrieval prediction of RDF graphs, respectively. *TP* represents the number of RDF graphs that the proposed approach predicts correctly as *hit*, *TN* represents the number of RDF graphs that the proposed approach predicts correctly as *miss*, *FP* represents the number of RDF graphs that the proposed approach predicts incorrectly as *hit*, and *FN* represents the number of RDF graphs that the proposed approach predicts incorrectly as *miss*.

3.4 Results

RQ1: Accuracy of the proposed approach. We answer the **RQ1** by performing a comparison between the proposed approach and the state-of-the-art approaches: MLSearch and GRSearch. We also compare the proposed approach with a random prediction algorithm and a zero-rule algorithm. We consider both algorithms because the proposed approach is the first approach to leverage deep learning algorithms for retrieval prediction of RDF graphs.

The evaluation results of the proposed approach and the baseline approaches are presented in Table 1. Approaches are presented in the first column of the table. The results of

Table 1. Comparison against baseline approaches.

	Acc	Pre	Rec	F1
Proposed Approach	97.12%	98.17%	95.56%	96.85%
MLSearch	85.21%	87.52%	79.19%	83.15%
GRSearch	78.63%	69.20%	66.31%	67.72%

https://doi.org/10.1371/journal.pone.0230500.t001

performance metrics (*Acc*, *Pre*, *Rec*, and *F1*) for each classifier are presented in Columns 2-5 of the table, respectively. Each row of the table presents the performance of the corresponding approach, respectively. The average *Acc*, *Pre*, *Rec*, and *F1* of the proposed approach, MLSearch, and GRSearch are (*97.12%*, *85.21%*, and *78.63%*), (*98.17%*, *87.52%*, and *69.20%*), (*95.56%*, *79.19%*, and *66.31%*), and (*96.85%*, *83.15%*, and *67.72*), respectively.

Table 2 shows the evaluation results of random prediction, zero-rule, and the proposed approach. Approaches are presented in the first column of the table. The results of performance metrics (*Acc*, *Pre*, *Rec*, and *F1*) for each classifier are presented in Columns 2-5 of the table, respectively. The rows of the table present the performance of the approaches, respectively. The average *Acc*, *Pre*, *Rec*, and *F1* of the proposed approach, random prediction, and zero-rule are (*97.12%*, *65.40%*, and *87.62%*), (*98.17%*, *65.93%*, and *80.12%*), (*95.56%*, *55.56%*, and *83.26%*), and (*96.85%*, *60.36%*, and *81.66*), respectively.

The observations from Tables 1 and 2 are as follows:

- The proposed approach outperforms the baseline approaches, random prediction, and zero-rule classifiers in *Acc*, *Pre*, *Rec*, and *F1*, respectively.
- The improvement of the proposed approach upon MLSearch in *Acc* and *F1* is *13.98% = (97.12%—85.21%) / 85.21%* and *16.52% = (96.85%—83.12%) / 83.12%*, respectively.
- The improvement of the proposed approach upon GRSearch in *Acc* and *F1* is *23.52% = (97.12%—78.63%) / 78.63%* and *43.02% = (96.85%—67.72%) / 67.72%*, respectively.
- The improvement in the performance of the proposed approach upon random prediction in *Acc* and *F1* is *40.86% = (92.12%—65.40%) / 65.40%* and *57.17% = (94.86%—60.36%) / 60.36%*, respectively.
- The improvement in the performance of the proposed approach upon zero-rule in *Acc* and *F1* is *5.13% = (92.12%—87.62%) / 87.62%* and *16.17% = (94.86%—81.66%) / 81.66%*, respectively.

We present the accuracy distribution of 10-fold cross-validation for the proposed approach and baseline approaches in Fig 7. We compare the *F1* distributions of each approach and plot one bean against each approach. Each short horizontal line within a bean illustrates the *F1* on a i^{th} fold, whereas the long horizontal line illustrates the average *F1*. We observe that the proposed approach outperforms the baseline approach in each fold. Notably, the average *F1* of the proposed approach is significantly large as compared to the best performances of the baseline approach.

Table 2. Comparison against random prediction and zero-rule.

	Acc	Pre	Rec	F1
Proposed Approach	97.12%	98.17%	95.56%	96.85%
Random Prediction	65.40%	65.93%	55.65%	60.36%
Zero-rule	87.62%	80.12%	83.26%	81.66%

https://doi.org/10.1371/journal.pone.0230500.t002

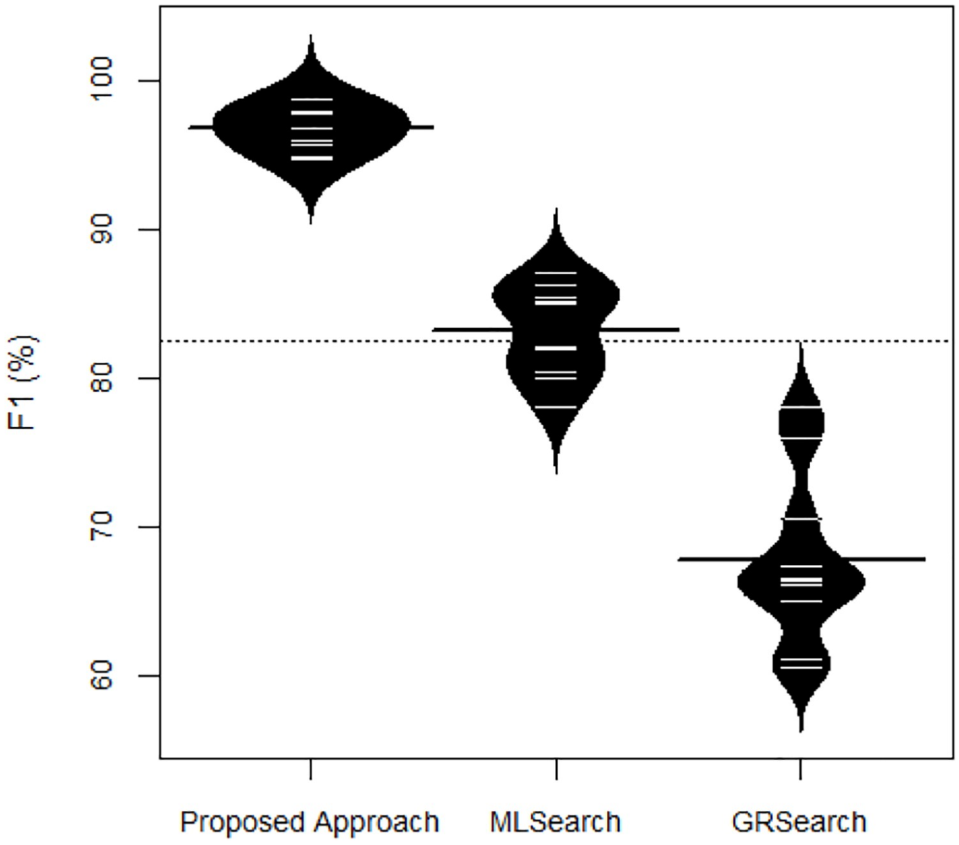


Fig 7. Distribution on accuracy.

https://doi.org/10.1371/journal.pone.0230500.g007

We also employ ANOVA (one-way) to confirm the significance of the proposed approach and the basline approach. It may examines whether the single factor (i.e., different approaches) is the only difference that drives to the difference in performance. Note that ANOVA is conducted independently on the *Acc*, *Pre*, *Rec*, and *F1*. Table 3 presents the results of ANOVA that shows $F > F_{cric}$ and $P_{Value} < (alpha = 0.05)$ are true for each *Acc*, *Pre*, *Rec*, and *F1*. It indicates that using different approach (the single factor) has a significant difference in the performances of both approaches. The preceding analysis concludes that the proposed approach is accurate in retrieval anticipation of RDF graphs.

RQ2: Performance comparison of machine/deep learning algorithms. We answer the **RQ2** by applying the most adopted machine/deep learning text classification algorithms (*CNN*, *LSTM*, *MNB*, *LR*, *RF*, and *SVM*) due to their competitive performance [27, 37–39]. The evaluation of the proposed approach with *SVM* yields most accurate results and outperforms the other classifiers on the given dataset.

Table 4 presents the evaluation results of *CNN*, *LSTM*, *SVM*, *LR*, *RF*, and *MNB*. The first column of the table presents the cross-validations. Columns 2-5 of the table present the performance results of *Acc*, *Pre*, *Rec*, and *F1* for each classifier, respectively. Rows of the table present the performance of a particular classifier, respectively.

The average *Acc*, *Pre*, *Rec*, and *F1* of *CNN*, *LSTM*, *SNM*, *MNB*, *LR*, and *RF* are (*97.12%*, *89.96%*, *85.21%*, *83.24%*, *90.54%*, and *91.33%*), (*98.17%*, *91.06%*, *87.52%*, *77.14%*, *93.69%*, and *92.27%*), (*95.56%*, *88.94%*, *79.19%*, *83.24%*, *95.15%*, and *94.88%*), and (*96.85%*, *89.99%*, *83.15%*, *80.07%*, *94.41%*, and *93.56*), respectively.

Table 3. Results of ANOVA analysis.

Source of Variation	SS	df	MS	F	p-value	F-crit
Acc						
Between Approaches	0.02777	1.00000	0.02777	26.54565	0.00007	4.41387
Within Approaches	0.01883	18	0.00105			
Total	0.04659	19				
Pre						
Between Approaches	0.01746	1	0.01746	30.57721	0.00003	4.41387
Within Approaches	0.01028	18	0.00057			
Total	0.02774	19				
Rec						
Between Approaches	0.02696	1	0.02696	17.56680	0.00055	4.41387
Within Approaches	0.02762	18	0.00153			
Total	0.05458	19				
F1						
Between Approaches	0.02212	1	0.02212	27.75361	0.00005	4.41387
Within Approaches	0.01435	18	0.00080			
Total	0.03647	19				

https://doi.org/10.1371/journal.pone.0230500.t003

The observations from the Table 4 are as follows:

- The *CNN* classifier surpasses all other classifiers in *accuracy*, *precision*, *recall* and *F1*. The reason is that *CNN* is better for extracting position invariant features as compared to *LSTM* and performs well with high dimensional feature sets [40].
- The *CNN* classifier surpasses all the other machine learning classifiers. It converts non-linearly classifiable and inter-dependent feature data into a higher-dimensional hyperplane if the classification of the data is not possible linearly.
- Although, the existing research [41] reports *MNB* classifier is effective in classification; however, it does not work well with the proposed approach on the given dataset. One possible reason is that the input predicates (features) to the classifier for training are inter-related, and *MNB* classifier performs well if the features are independent [27, 42]. The evaluation results of *MNB* on the given dataset are not effective as compared to *SVM*, *LR*, and *RF* with the proposed approach.
- The performance results of *LR* and *RF* are very close to the *SVM*. A larger dataset may reveal that one of them is better than *SVM*.

Table 4. Comparison against machine learning algorithms.

ML Classifier	Acc	Pre	Rec	F1
CNN	97.12%	98.17%	95.56%	96.85%
LSTM	89.96%	91.06%	88.94%	89.99%
SVM	85.21%	87.52%	79.19%	83.15%
MNB	83.24%	77.14%	83.24%	80.07%
LR	90.54%	93.69%	95.15%	94.41%
RF	91.33%	92.27%	94.88%	93.56%

https://doi.org/10.1371/journal.pone.0230500.t004

The preceding analysis concludes that *CNN* works better than the other classifiers with the proposed approach.

RQ3: Influence of features' preprocessing. The different RDF graphs may have similar predicates (features) or may have superlative/comparative words in predicates. Passing such data as features to a machine learning algorithm is an overhead. It reduces performance and increases the computational cost of machine learning algorithms.

We answer the **RQ3** by performing the comparison between the evaluation results of the proposed approach with and without features' preprocessing. The evaluation results are presented in Table 5. The preprocessing input settings are presented in the first column of the table. Columns 2-5 of the table present the performance results of *Acc*, *Pre*, *Rec*, and *F1*. The rows of the table present the performance of the proposed approach to the different settings of preprocessing, respectively. The improvement in the performance of the proposed approach with different preprocessing settings is presented in the last row of the table.

From the Table 5, we make the following observations:

- The preprocessing enabled proposed approach achieves significant improvement in performance. The evaluation results suggest that the performance improvement in *Acc*, *Pre*, *Rec*, and *F1* are up to *16.71%*, *14.38%*, *11.22%*, and *12.82%*, respectively.
- The preprocessing disabled approach significantly decreases the *Rec* from *95.56%* to *85.92%*. The decrease in *Rec* returns incorrect results against the requested query. One possible reason of the decrease in performance is the similar or superlative/comparitive words in the predicates of the given triples.

The preceding analysis concludes that preprocessing of the features is essential to the proposed approach.

3.5 Threats to validity

There could be some elements that may affect the performance of the proposed approach. The followings are the threats to the validity of the proposed approach.

- The selection of evaluation metrics is the first threat to construct validity. We select *Acc*, *Pre*, *Rec*, and *F1* metrics for the evaluation of the proposed approach. Because, they are the most adopted metrics [32–37] for the evaluation of classification problems.
- The leverage of *NLTK* for the preprocessing of the extracted features (as mentioned in Section 2.5) is a threat to construct validity. We select *NLTK* due to its performance and popularity [37]. The use of any other natural language processing repository may affect the said results of the proposed approach.
- The generalizability of the proposed approach is a threat to external validity. We focus the RDF graphs from an open-source dataset (*DBpedia*) for the evaluation of the proposed approach. We cannot guarantee the results of the proposed approach with other datasets.

Table 5. Influence of preprocessing.

Preprocessing	Acc	Pre	Rec	F1
Enable	92.12%	94.17%	95.56%	94.86%
Disable	78.93%	82.33%	85.92%	84.08%
Improvement	**16.71%**	**14.38%**	**11.22%**	**12.82%**

https://doi.org/10.1371/journal.pone.0230500.t005

4 Related work

The WWW is an information space where RDF graphs and other web resources are identified by URLs that may be interlinked and are accessible over the Internet. It is difficult to get the right URLs against asked queries due to the information overload caused by the current digital era. To address this problem, Tim Burner Lee introduced the semantic web that provides a common framework and allows data to be shared and reused across applications. It considers semantics for searching rather than keyword matching and query responses. Linking data together from different resources is the key to the semantic web. Moreover, linking data is essential to connect and search data over the semantic web. Linked data rely on RDF graphs that contain data in RDF format. Many approaches have been proposed on the efficient search of RDF graphs. Such approaches mainly focus on classical RDF searching e.g., keyword-based searching or graph-based searching.

Tran et al. [43] introduced the idea of generating summary-graphs for the original RDF graph to generate and rank candidate SPARQL queries. Then, Zhang et al. [44] proposed a solution to this idea. Moreover, Yang et al. [45] proposed tree patterns to connect keywords specified by the users where the tree patterns are ordered by their size relevance, and Zheng et al. [46] proposed a method to search semantically equivalent structure patterns. Finally, De Virgilio [47] proposed an RDF keyword-based query vis Tensor calculus and later extended it to a distributed environment via MapReduce [48].

Nagarajan *et al.* [49] presented ontology-based multi-model semantic information retrieval system. It is based on the idea of integrating domain knowledge and images and retrieves the required multi-modal information using a fuzzy rule set. It also provides the image semantic by constructing visual words using the probabilistic latent semantic. Other researches [50–52] also proposed formalize and semantic visualization models based on the fuzzy rule set.

Nhuan *et al.* [53] proposed an approach that determines the degrees of equality between relations (properties) defined by different vocabularies. They consider the occurrences of matching pairs of RDF triples to find the intervals representing lower and upper levels of property equality. Consequently, they obtained a graph of similar properties where the interval-based strength of edges represents degrees of similarity between properties.

Jaafar *et al.* [54] proposed a fuzzy knowledge-based framework to realize a nature and visualized F-RDF retrieval operation, to help an end-user to enhance the querying and accessing Web data.

Gupta *et al.* [18] introduced a ranking function based on fuzzy logic to enhance Information Retrieval. The function based on the computation of term-weighting schemas such as term frequency, inverse document frequency, and normalization. The state-of-the-art [15–18] has described the difficulties in the understanding of a semantic search engine. The motive behind is to propose an approach based on RDF, and the automatic identification of content over the WWW.

As a conclusion, researchers have proposed different approaches [8–19, 55, 56] for retrieving information using RDF; however, it requires significant improvement. Moreover, none of them employs machine learning classification algorithms to address this problem. Notably, the proposed approach differs in that the existing approaches as we are first to apply the support vector machine for the retrieval of RDF graphs.

5 Conclusion

In this digital era, Web users share almost every moment of daily life on the Internet that causes information overload. Consequently, it is difficult to accurately retrieve the required information without understanding the syntax and semantics of the content. To this end, in this paper, a deep learning-based approach for searching RDF graphs is proposed that treats RDF graph requests as a classification problem. The proposed approach applies a deep learning classifier on the given dataset for the retrieval anticipation of RDF graphs. The proposed approach introduces a new way to search the RDF graphs and helps the Web users in answering their queries. We perform the 10-fold cross-validation for the evaluation of the proposed approach using the open-source RDF graphs of *DBpedia*. The evaluation results show that the proposed approach is accurate.

The broader impact of this study is to indicate that the triples in the RDF graphs are a rich source of information for accurate retrieval prediction of RDF graphs. Our results motivate future research on the retrieval anticipation of RDF graphs. We want to investigate a retrieval prediction of RDF graphs with a deep learning approach with deep hyperparameter settings. This will also confirm the generalizability of the proposed approach.

Author Contributions

Conceptualization: Hatem Soliman.

Formal analysis: Hatem Soliman.

Investigation: Hatem Soliman.

Methodology: Hatem Soliman.

Validation: Hatem Soliman.

Writing – original draft: Hatem Soliman.

References

1. Zhou L, Ding L, Finin T. How is the Semantic Web evolving? A dynamic social network perspective. Computers in Human Behavior. 2011; 27:1294–1302. https://doi.org/10.1016/j.chb.2010.07.024
2. Lv Y, Ma Z, Yan l. Fuzzy RDF: A Data Model to Represent Fuzzy Metadata; 2008. p. 1439–1445.
3. Ma Z, Yan l. Modeling fuzzy data with RDF and fuzzy relational database models. International Journal of Intelligent Systems. 2018; 33. https://doi.org/10.1002/int.21996
4. Li G, Yan L, Ma Z. An approach for approximate subgraph matching in fuzzy RDF graph. Fuzzy Sets and Systems. 2019; 376:106–126. https://doi.org/10.1016/j.fss.2019.02.021.
5. Bizer C, et al. Linked Data—The story so far; 2009.
6. Bizer C. The Emerging Web of Linked Data. IEEE Intelligent Systems. 2009; 24(5):87–92. https://doi.org/10.1109/MIS.2009.102
7. Casanova MA. Keyword Search over RDF Datasets. In: Laender AHF, Pernici B, Lim EP, de Oliveira JPM, editors. Conceptual Modeling. Cham: Springer International Publishing; 2019. p. 7–10.
8. Singh V, Zong B, Singh AK. Nearest Keyword Set Search in Multi-Dimensional Datasets. IEEE Transactions on Knowledge and Data Engineering. 2016; 28(3):741–755. https://doi.org/10.1109/TKDE.2015.2492549
9. Gani A, Siddiqa A, Shamshirband S, Nasaruddin F. A survey on Indexing Techniques for Big Data: Taxonomy and Performance Evaluation. Knowledge and Information Systems. 2015; 46.
10. Elleuch N, Zarka M, Anis Ba, Alimi A. A fuzzy ontology—Based framework for reasoning in visual video content analysis and indexing. Proceedings of the 11th International Workshop on Multimedia Data Mining, MDMKDD'11—Held in Conjunction with SIGKDD'11. 2011.
11. Gacto M, Alcalá R, Herrera F. Integration of an Index to Preserve the Semantic Interpretability in the Multiobjective Evolutionary Rule Selection and Tuning of Linguistic Fuzzy Systems. Fuzzy Systems, IEEE Transactions on. 2010; 18:515–531. https://doi.org/10.1109/TFUZZ.2010.2041008
12. Komkhao M, Lu J, Li Z, Halang WA. Incremental collaborative filtering based on Mahalanobis distance and fuzzy membership for recommender systems. International Journal of General Systems—INT J GEN SYSTEM. 2012; 42:1–26.
13. Li W, X Chen C. Efficient data modeling and querying system for multi-dimensional spatial data; 2008. p. 58.

14. Izakian H, Pedrycz W, Jamal I. Fuzzy clustering of time series data using dynamic time warping distance. Engineering Applications of Artificial Intelligence. 2015; 39:235–244. https://doi.org/10.1016/j.engappai.2014.12.015.

15. Lughofer E, Pratama M. Online Active Learning in Data Stream Regression Using Uncertainty Sampling Based on Evolving Generalized Fuzzy Models. IEEE Transactions on Fuzzy Systems. 2018; 26 (1):292–309. https://doi.org/10.1109/TFUZZ.2017.2654504

16. Idreos S, Papaemmanouil O, Chaudhuri S. Overview of Data Exploration Techniques. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. SIGMOD'15. New York, NY, USA: ACM; 2015. p. 277–281. Available from: http://doi.acm.org/10.1145/2723372.2731084.

17. Reh A, Amirkhanov A, Kastner J, Groller E, Heinzl C. Fuzzy feature tracking: Visual analysis of industrial 4D-XCT data. Computers and Graphics. 2015; 53:177–184. https://doi.org/10.1016/j.cag.2015.04.001.

18. Gupta Y, Saini A, Saxena AK. A new fuzzy logic based ranking function for efficient Information Retrieval system. Expert Systems with Applications. 2015; 42(3):1223–1234. https://doi.org/10.1016/j.eswa.2014.09.009.

19. Arnaout H, Elbassuoni S. Effective searching of RDF knowledge graphs. Journal of Web Semantics. 2018; 48:66–84. https://doi.org/10.1016/j.websem.2017.12.001.

20. Hadi AS, Fergus P, Dobbins C, Al-Bakry AM. A Machine Learning Algorithm for Searching Vectorised RDF Data. In: 2013 27th International Conference on Advanced Information Networking and Applications Workshops; 2013. p. 613–618.

21. Anvik J, Murphy GC. Reducing the Effort of Bug Report Triage: Recommenders for Development-oriented Decisions. ACM Trans Softw Eng Methodol. 2011; 20(3):10:1–10:35. https://doi.org/10.1145/2000791.2000794

22. Hindle A, Barr ET, Su Z, Gabel M, Devanbu P. On the Naturalness of Software. In: Proceedings of the 34th International Conference on Software Engineering. ICSE'12. Piscataway, NJ, USA: IEEE Press; 2012. p. 837–847. Available from: http://dl.acm.org/citation.cfm?id=2337223.2337322.

23. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed Representations of Words and Phrases and Their Compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems—Volume 2. NIPS'13. USA: Curran Associates Inc.; 2013. p. 3111–3119. Available from: http://dl.acm.org/citation.cfm?id=2999792.2999959.

24. Hochreiter S, Schmidhuber J. Long Short-term Memory. Neural computation. 1997; 9:1735–80. https://doi.org/10.1162/neco.1997.9.8.1735 PMID: 9377276

25. Sak H, Senior A, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. 2014; p. 338–342.

26. Keras; 2015. https://keras.io/.

27. Ramay WY, Umer Q, Yin XC, Zhu C, Illahi I. Deep Neural Network-Based Severity Prediction of Bug Reports. IEEE Access. 2019; 7:46846–46857. https://doi.org/10.1109/ACCESS.2019.2909746

28. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R. Improving neural networks by preventing co-adaptation of feature detectors. 2012.

29. Keras. Flatten Layer;. Retrieved Nov 01, 2018 from https://github.com/keras-team/keras/blob/master/keras/layers/core.py#L467.

30. Gayathri P, Rajendran VV. Semantic search on summarized RDF triples. In: 2017 International Conference on Intelligent Computing and Control (I2C2); 2017. p. 1–6.

31. Purushotham S, Tripathy BK. Evaluation of Classifier Models Using Stratified Tenfold Cross Validation Techniques. In: Krishna PV, Babu MR, Ariwa E, editors. Global Trends in Information Systems and Software Applications. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. p. 680–690.

32. Tian Y, Lo D, Sun C. Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction. In: Proceedings of the 2012 19th Working Conference on Reverse Engineering. WCRE'12. Washington, DC, USA: IEEE Computer Society; 2012. p. 215–224. Available from: http://dx.doi.org/10.1109/WCRE.2012.31.

33. Tian Y, Lo D, Sun C. DRONE: Predicting Priority of Reported Bugs by Multi-factor Analysis. In: 2013 IEEE International Conference on Software Maintenance; 2013. p. 200–209.

34. Lamkanfi A, Demeyer S, Giger E, Goethals B. Predicting the severity of a reported bug. In: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010); 2010. p. 1–10.

35. Lamkanfi A, Demeyer S, Soetens QD, Verdonck T. Comparing Mining Algorithms for Predicting the Severity of a Reported Bug. In: 2011 15th European Conference on Software Maintenance and Reengineering; 2011. p. 249–258.

36. Yang G, Baek S, Lee JW, Lee B. Analyzing Emotion Words to Predict Severity of Software Bugs: A Case Study of Open Source Projects. In: Proceedings of the Symposium on Applied Computing. SAC'17. New York, NY, USA: ACM; 2017. p. 1280–1287. Available from: http://doi.acm.org/10.1145/3019612.3019788.

37. Umer Q, Liu H, Sultan Y. Emotion Based Automated Priority Prediction for Bug Reports. IEEE Access. 2018; 6:35743–35752. https://doi.org/10.1109/ACCESS.2018.2850910

38. Wu X, Kumar V, Ross Quinlan J, Ghosh J, Yang Q, Motoda H, et al. Top 10 algorithms in data mining. Knowledge and Information Systems. 2008; 14(1):1–37. https://doi.org/10.1007/s10115-007-0114-2

39. Sohrawardi SJ, Azam I, Hosain S. A comparative study of text classification algorithms on user submitted bug reports. In: Ninth International Conference on Digital Information Management (ICDIM 2014); 2014. p. 242–247.

40. Wang B. Disconnected Recurrent Neural Networks for Text Categorization. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics; 2018. p. 2311–2320. Available from: https://www.aclweb.org/anthology/P18-1215.

41. Hellerstein J, Jayram Ts, Rish I. Recognizing End-User Transactions in Performance Management.; 2000. p. 596–602.

42. Umer Q, Liu H, Sultan Y. Sentiment based approval prediction for enhancement reports. Journal of Systems and Software. 2019; 155:57–69. https://doi.org/10.1016/j.jss.2019.05.026.

43. Tran T, Wang H, Rudolph S, Cimiano P. Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In: Proceedings of the 2009 IEEE International Conference on Data Engineering. ICDE'09. Washington, DC, USA: IEEE Computer Society; 2009. p. 405–416. Available from: http://dx.doi.org/10.1109/ICDE.2009.119.

44. Zhang L, Tran T, Rettinger A. Probabilistic Query Rewriting for Efficient and Effective Keyword Search on Graph Data. Proc VLDB Endow. 2013; 6(14):1642–1653. https://doi.org/10.14778/2556549.2556550

45. Yang M, Ding B, Chaudhuri S, Chakrabarti K. Finding Patterns in a Knowledge Base Using Keywords to Compose Table Answers. Proc VLDB Endow. 2014; 7(14):1809–1820. https://doi.org/10.14778/2733085.2733088

46. Zheng W, Zou L, Peng W, Yan X, Song S, Zhao D. Semantic SPARQL Similarity Search over RDF Knowledge Graphs. Proc VLDB Endow. 2016; 9(11):840–851. https://doi.org/10.14778/2983200.2983201

47. De Virgilio R. RDF Keyword Search Query Processing via Tensor Calculus. vol. 7566; 2012. p. 780–788.

48. De Virgilio R, Maccioni A. Distributed Keyword Search over RDF via MapReduce. In: Presutti V, d'Amato C, Gandon F, d'Aquin M, Staab S, Tordai A, editors. The Semantic Web: Trends and Challenges. Cham: Springer International Publishing; 2014. p. 208–223.

49. Nagarajan G, Minu RI. Fuzzy Ontology Based Multi-Modal Semantic Information Retrieval. Procedia Computer Science. 2015; 48:101–106. https://doi.org/10.1016/j.procs.2015.04.157.

50. Dong F, Hirota K. In: Tamir DE, Rishe ND, Kandel A, editors. Formalization and Visualization of Kansei Information Based on Fuzzy Set Approach. Cham: Springer International Publishing; 2015. p. 169–181.

51. Pancho DP, Alonso JM, Magdalena L. Enhancing Fingrams to deal with precise fuzzy systems. Fuzzy Sets and Systems. 2016; 297:1–25. https://doi.org/10.1016/j.fss.2015.05.019.

52. Besbes G, Zghal H. Personalized and context-aware retrieval based on fuzzy ontology profiling. Integrated Computer Aided Engineering. 2016; 24. https://doi.org/10.3233/ICA-160525

53. To ND, Reformat MZ, Yager RR. Linked Open Data: Uncertainty in Equivalence of Properties. In: Kacprzyk J, Szmidt E, Zadrożny S, Atanassov KT, Krawczak M, editors. Advances in Fuzzy Logic and Technology 2017. Cham: Springer International Publishing; 2018. p. 418–429.

54. Jaafar J, Danyaro KU, Liew MS. Web Intelligence: A Fuzzy Knowledge-Based Framework for the Enhancement of Querying and Accessing Web Data. In: Handbook of Research on Trends and Future Directions in Big Data and Web Intelligence; 2015. p. 83–104.

55. Kyu KM, Oo AN. Graph-based Indexing Method for Searching in RDF Data. In: 2019 International Conference on Advanced Information Technologies (ICAIT); 2019. p. 96–101.

56. Gayathri P, Rajendran VV. Semantic search on summarized RDF triples. In: 2017 International Conference on Intelligent Computing and Control (I2C2); 2017. p. 1–6.